Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rok Klemen

# Aplikacija za prikaz prostorskih podatkov

diplomsko delo

visokošolski strokovni študijski program
računalništva in informatike

Ljubljana, 2016

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Rok Klemen

# Aplikacija za prikaz prostorskih podatkov

diplomsko delo

visokošolski strokovni študijski program
računalništva in informatike

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2016

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Različni tipi aplikacij potrebuje možnost prikaza prostorskih podatkov. Zato je smiselno razviti aplikacijo, ki bo le prikazovala prostorske podatke in bo omogočala enostaven način integracije z različnimi aplikacijami.

Naredite analizo in načrt za takšno aplikacijo. Aplikacija naj se osredotoči na branje prostorskih podatkov iz datoteke SHP, ter prikaz teh podatkov v programu. Po izdelanem načrtu pa aplikacijo tudi razvijte v programskem jeziku C++, grafični uporabniški vmesnik pa naj bo izdelan s pomočjo spletnih tehnologij (HTML, JavaScript, JQuery, JQuery UI). Povezanost med temi tehnologijami pa naj bo dosežena s pomočjo razvojnega ogrodja QT.

# Izjava o avtorstvu diplomskega dela

Spodaj podpisani Rok Klemen sem avtor diplomskega dela z naslovom:

*Aplikacija za prikaz prostorskih podatkov*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Roka Rupnika,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 4. februarja 2016 Podpis avtorja: _____

*Zahvaljujem se mentorju doc. dr. Roku Rupniku za potrpežljivost, usmerjanje in strokovno pomoč pri pisanju diplomske naloge.*

*Simonu se zahvaljujem za podporo in vso posredovano znanje, brez katerega izdelava diplomske naloge ne bi bila mogoča, ter Marku za pomoč med študijem.*

*Posebno zahvalo bi rad naklonil staršema in bratu za vso podporo v času študija.*

# Kazalo

# Seznam uporabljenih kratic

| kratica | angleško | slovensko |
|---|---|---|
| **API** | Application Programming Interface | vmesnik za programiranje |
| **AJAX** | Asynchronous JavaScript and XML | tehnologija za ustvarjanje dinamičnih spletnih aplikacij in strani |
| **CAD** | Computer Aided Design | računalniško podprto načrtovanje |
| **CSS** | Cascading Style Sheets | slogovna predloga, ki določa izgled spletnih strani |
| **ESRI** | Environmental Systems Research Institute | inštitut za raziskavo okoljskih sistemov |
| **FDO** | Feature Data Object | zbirka geometrijskih objektov, s pripojenim opisom geoprostorkih lastnosti |
| **GIS** | Geographic Information System | geografski informacijski sistem |
| **HTML** | HyperText Markup Language | označevalni jezik za izdelavo spletnih strani |
| **MFC** | Microsoft Foundation Class Library | knjižnica, ki omogoča razvoj aplikacij operacijskega sistema Windows v programskem jeziku C++ |

| | | |
|---|---|---|
| **NSIS** | Nullsoft Scriptable Install System | skriptni namestitveni sistem podjetja Nullsoft, Inc. |
| **OSGeo** | Open Source Geospatial Foundation | neprofitna organizacija, ki skrbi za razvoj odprto kodne geoprostorske programske opreme |
| **SDK** | Software Development Kit | paket za razvoj programske opreme |
| **SHP** | Esri Shapefile | tip geoprostorske datoteke, ki ga je razvil inštitut ESRI |
| **XML** | Extensible Markup Language | razširljiv jezik za definiranje strukture podatkov in dokumentov |

# Povzetek

**Naslov:** Aplikacija za prikaz prostorskih podatkov

Namen diplomske naloge je razvoj aplikacije, ki uporabniku omogoča prikaz prostorskih podatkov v programu/sistemu CAD. Aplikacija se osredotoči na branje prostorskih podatkov iz datoteke SHP, ter prikaz teh podatkov v programu BricsCAD. V nalogi se opredeli pojem CAD, predstavi nekatere programe/sisteme CAD ter datoteka SHP in njene značilnosti. Naloga predstavi, zakaj je takšna aplikacija potrebna in komu je namenjena. Osrčje aplikacije je razvito v programskem jeziku C++, grafični uporabniški vmesnik pa je izdelan s pomočjo spletnih tehnologij (HTML, JavaScript, JQuery, JQuery UI). Povezanost med temi tehnologijami je dosežena s pomočjo razvojnega ogrodja QT. Diplomska naloga zajema tudi opis orodij in tehnologij, uporabljenih pri razvoju aplikacije.

**Ključne besede:** prostorski podatki, CAD, GIS, datoteka SHP, BricsCAD, AutoCAD, Qt, C++, HTML, Javascript.

# Abstract

**Title:** Aplication for viewing spatial data

The purpose of this thesis is to develop an application which allows the user to display spatial data in a CAD system/program. The application focuses on reading spatial data from a SHP file and displaying the data in BricsCAD. The thesis defines the concept of CAD, presents some CAD programs/systems, and explains the SHP file as well as its features. It explains why such an application is needed and who the target audience is. The central part of the application is developed in C++, while the graphical user interface is developed using web technologies (HTML, JavaScript, JQuery, JQuery UI). The connection between these two different technologies is achieved with the help of the QT development framework. The thesis also includes a description of the tools and technologies used in the development of the application.

**Keywords:** spatial data, CAD, GIS, SHP file, BricsCAD, AutoCAD, Qt, C++, HTML, Javascript.

# Poglavje 1

# Uvod

V svetu gradbeništva, arhitekture in ostalih inženirskih strok je princip načrtovanja od začetka osemdesetih let tesno povezan z računalništvom. Kratica CAD oziroma Computer Aided Design se uporablja za zelo veliko skupino programov, ki omogočajo vektorsko načrtovanje, integrirano v računalniško okolje.

Če je bilo v začetku tega obdobja najpomembnejše izdelati ustrezen načrt, ki je prikazoval tlorise, prereze in ostale prikaze načrtovanih objektov, je v nadaljevanju razvoja stroke postalo ravno tako pomembno prostorsko umeščanje teh objektov že v fazi samega načrtovanja.

Po drugi strani se je tudi razvoj geografskih informacijskih sistemov (GIS) preselil v računalniška okolja že globoko v začetku šestdesetih let prejšnjega stoletja. Do današnjih dni so se sistemi GIS v računalniškem svetu, sploh pa v spletnih okoljih, naselili pravzaprav v vsa področja podajanja informacij, kjer smo kakor koli povezani s podajanjem lokacije. Zato je velika večina držav in njihovih podrejenih ustanov, ki se ukvarjajo s popisom in prikazom sveta, zgrajenega okoli nas, v ta namen privzela eno od oblik sistemov GIS, ki jim služijo za stvarni prikaz teh baz podatkov.

Če smo prej omenili pomembnost prostorskega umeščanja, se sedaj srečujemo z vprašanjem povzemanja in prikazovanja podatkov iz GIS-a v okolju CAD, ki bi omogočilo čim bolj natančno umestitev načrtovanih objektov,

na primer cest, hiš in kanalizacij, v obstoječe okolje. To umestitev morata gradbenik ali arhitekt izvesti tako, da ne prihaja do težav med novimi in obstoječimi objekti v času gradnje.

Takšno umeščanje je dokaj uspešno rešilo podjetje Autodesk s svojo aplikacijo AutoCAD Map 3D, ki omogoča črpanje podatkov iz sistemov GIS. Hkrati je njena osnova AutoCAD, ki predstavlja najbolj razširjen program CAD in se ga uporablja v gradbeništvu, arhitekturi, strojništvu in vseh drugih inženirskih strokah. Težava pri vseh proizvodih podjetja Autodesk je njihova izredno visoka cena.

Zaradi hude krize v gradbeništvu je bilo vse več projektantov prisiljeno v iskanje cenovno bolj ugodnih možnosti, s čimer se je odprlo tržišče konkurenčnim razvijalcem programov CAD, kot sta kitajski ZWSOFT s programom ZWCAD+ in belgijski Bricsys s programom BricsCAD. Oba programa sta primerljiva z osnovnim programom AutoCAD, ne pa tudi s programom AutoCAD Map 3D, saj nimata razvitih nobenih funkcionalnosti, ki bi omogočale povezavo s sistemi GIS. Vrednost teh programov bi se izjemno povečala že z vsako manjšo funkcionalnostjo, ki bi omogočila vnos podatkov GIS v okolje osnovnega programa CAD.

# Poglavje 2

# CAD (Računalniško podprto načrtovanje)

## 2.1 Kaj je CAD

CAD oziroma Computer-Aided Design predstavlja vektorsko podprto vizualno 3D-modeliranje oziroma načrtovanje s pomočjo računalnika.

Izraz Computer-Aided Design oz. kratica CAD izhaja iz istoimenskega projekta, ki so ga v poznih petdesetih in začetku šestdesetih izvajali na inštitutu MIT v ZDA. Kot sam avtor izraza je naveden Douglas T. Ross, ki je bil tudi vodja tega projekta [2]. Kot zanimivost – Douglas T. Ross je dobil zamisel za CAD ob opazovanju radarskih naprav, ki so preračunavale vektorje poti letal. Dobro desetletje kasneje je iz omenjenega projekta nastal sistem SKETCHPAD, ki ga lahko štejemo za pionirskega med programi za vizualno 3D-modeliranje na računalnikih in prednika vseh današnjih sistemov CAD [2]. Sicer bi lahko na tem mestu našteli še nekaj programov, ki so se razvili vzporedno, toda omenjeni je bil najnaprednejši.

V sedemdesetih letih so računalniki postajali vse bolj dostopni in vzporedno so se razvijali tudi sistemi CAD. Programi, ki so nastali na področju CAD-a so imeli skupno vizijo – kako čim bolje podpreti inženirje pri njihovem delu. Kljub skupni viziji so bile ideje in poti različne, leta 1977 pa je

bil razvit program Interact CAD, iz katerega je pet let kasneje nastal Autodeskov AutoCAD (AutoCAD 1.0 je luč sveta ugledal decembra leta 1982). Prva različica je poznala osnovne objekte, ki jih vsebuje še danes - 31 let in 28 generacij kasneje: to so bile črte, krogi, loki in tekst. Z njimi se lahko oblikujejo vsi kompleksnejši objekti [2].

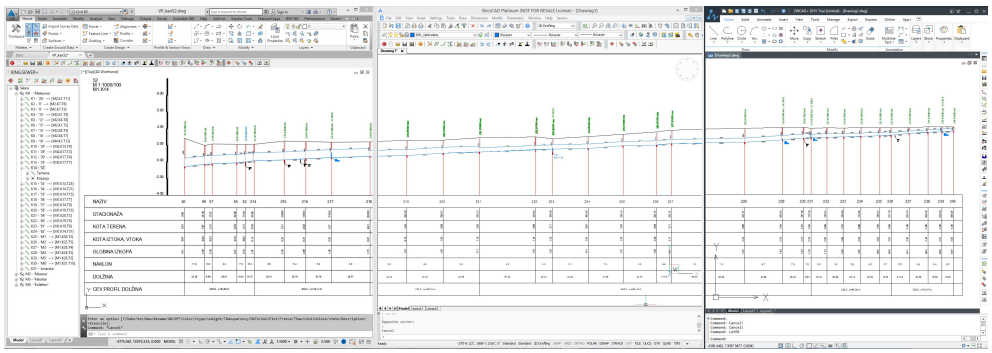
## 2.2 Autodesk in AutoCAD

AutoCAD je do današnjega dne najbolj široko uporabljan program CAD na svetu. Med tekmeci iz istega časovnega okvira s popolnoma lastnim okoljem se lahko omeni Bentley Microstation. Poleg tega je nastalo še nekaj programov, ki so bili sprva razviti za lastne potrebe nekaterih tovarn, npr. CATIA v francoski firmi Dassault Systemes (znani po lovskih letalih Mirage), in jih je prosti trg spoznal šele kasneje. Sčasoma je AutoCAD postal standardno orodje za projektiranje CAD. Število prodanih licenc je že leta 2007 doseglo 8 milijonov. Drugo s tem povezano dejstvo je, da sta oba osnovna formata, ki jih podpira AutoCAD, to sta DXF in DWG, postala standardna formata za medobratovalnost med programi CAD. Format DXF je možno prebrati tudi v formi ASCII, vendar ima nekatere omejitve pri tipih objektov, ki jih DWG nima.

Autodesk je zaradi formata DWG vložil vrsto tožb proti drugim podjetjem in neprofitnim organizacijam z namenom, da DWG zaščiti kot lastno blagovno znamko. Med pomembnejšimi sodnimi bitkami je bila tožba proti neprofitni organizaciji Open Design Alliance. Končni rezultat je bil, da je Autodesk izgubil tožbo, saj je ameriški patentni urad (USPTO) odločil, da format DWG ne more biti blagovna znamka. Ta odločitev je izjemnega pomena za odprto širjenje risb CAD v tem formatu in je osnova za lažji razvoj nizkocenovnih orodij CAD, ki so v svoji osnovi kloni programa AutoCAD.

Zaradi svoje, skoraj monopolne vloge na tržišču, si je Autodesk lahko privoščil, da je ceno svojih proizvodov držal zelo visoko. Tako je trenutno cena osnovne različice AutoCAD-a v EU (in s tem tudi v Sloveniji) 5.000,00

EUR, kar je dodatnih 33 % višje od cene na ameriškem tržišču (podatek pridobljen od uradnega zastopnika za Autodesk v Sloveniji: podjetje SL-King d.o.o. iz Ljubljane). Glede na gospodarsko krizo je ta cena sedaj velika ovira za nakup licenčne programske opreme CAD za večino projektivnih birojev, zato so se začeli ozirati tudi po drugih možnostih. Z odpravo možnosti zaščite formata DWG kot blagovne znamke, s pravim pristopom k neodvisnemu tehnološkemu napredku in s skoraj neprimerljivo nižjo ceno so se odprla vrata za nekatere od nizkocenovnih programov CAD, ki so, vsaj po svojem videzu, še vedno bolj ali manj natančne kopije programa AutoCAD. Za uspešnost teh alternativnih proizvodov je potrebna še ena prvina – odprta platforma za dodatna programska orodja, ki jih lahko programerji razvijajo v jezikih, kot sta C++ ali .NET. Ob pregledu tržišča sta se izkazala kot najboljša izbira dva proizvoda – ZWCAD+ kitajskega podjetja ZWSOFT in BricsCAD belgijskega podjetja BRICSYS.



Slika 2.1: Prikaz aplikacije Sewer+ v programih AutoCAD, BricsCAD in ZwCAD. Aplikacija deluje v vseh treh programih.

## 2.3 BricsCAD

BricsCAD je program, razvit v podjetju Bricsys, ki je bilo ustanovljeno leta 2002 v Belgiji. Program deluje v operacijskih sistemih Windows, Linux in Mac.

BricsCAD za branje in pisanje datotek DWG uporablja knjižnice DWG Open Design Alliance. Za pisanje programov za BricsCAD se lahko uporabljajo skripte AutoLISP, DCL, VBA in BRX. Iz funkcionalnega vidika je BricsCAD poravnan z najsodobnejšimi funkcionalnostmi, ki jih vsebuje program Autodesk AutoCAD. V generaciji 12 iz leta 2011 so mu dodali 3D-modeliranje in v naslednjih generacijah je dobil še možnost parametričnega modeliranja. Najnovejša različica je V16, ki je bila izdana konec oktobra 2015, ima kot novost vključene tudi funkcionalnosti BIM [1].

Cenovno je BricsCAD bistveno ugodnejši od AutoCAD-a, saj je najzmogljivejša različica Platinum dosegljiva za ceno 1.000,00 EUR (podatek pridobljen od uradnega zastopnika za Bricsys v Sloveniji: podjetje SL-King d.o.o. iz Ljubljane). SDK, ki se uporablja za pisanje programov za BricsCAD, je izjemno zmogljiv in omogoča kvalitetno izdelavo specializiranih rešitev, kot je tudi aplikacija, opisana v tej diplomski nalogi.

# Poglavje 3

# Geografski informacijski sistemi

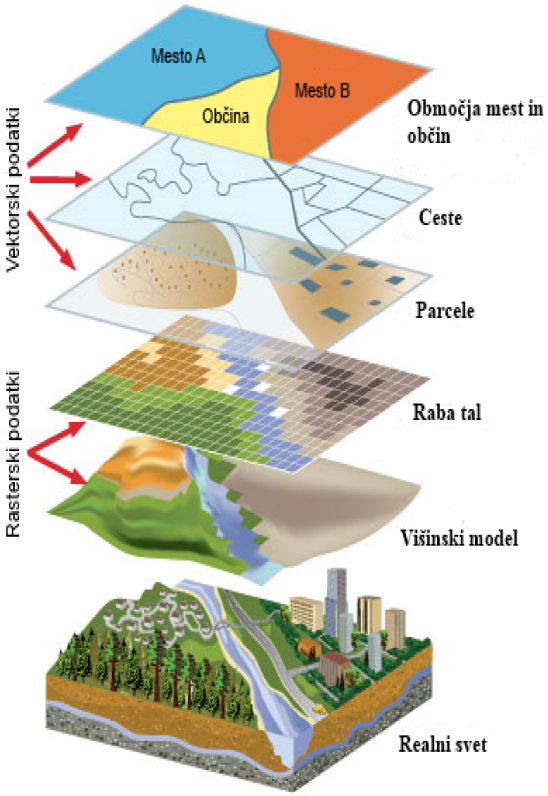
## 3.1 Kaj je geografski informacijski sistem

Geografski informacijski sistem (GIS) v strogem smislu zajema vsak informacijski sistem, ki integrira, shranjuje, ureja, analizira, deli in prikazuje geografske podatke. V bolj splošnem pomenu GIS predstavljajo orodja, ki uporabniku omogočajo ustvarjanje interaktivnih poizvedb, analiziranje prostorski podatkov, urejanje podatkov, kart in prikazovanje rezultatov vseh teh operacij [7].

Geografski informacijski sistem tako zajema skupek strojne opreme, programske opreme in postopkov, ki omogočajo zajemanje, upravljanje, analiziranje, modeliranje in prikaz geografsko referenciranih podatkov na tako imenovanih "pametnih"kartah. To uporabnikom omogoča lažjo predstavo, razumevanje ter analizo vzorcev in odnosov med podatki [8].

## 3.2 GIS v gradbeništvu

GIS se je v današnjem času dotaknil že dejansko vseh področij našega življenja, saj je njegovo uporabo moč zaznati v geografskem kartiranju, beleženju klimatologije, telekomunikacijah, postopkih javne uprave, energetiki, javnih komunalnih službah, rudarstvu, letalstvu, ribolovu, javnem cestnem




Slika 3.1: Prikaz slojev, ki lahko sestavljajo GIS [15].

prometu, itd.

Morda najširše uporabljena variacija GIS-a je spletno prikazovanje različnih tipov podatkov v prekrivajočih se slojih, pri čemer osnovni sloj predstavlja rastrska karta – zemljevid ali ortofoto posnetek. Višji sloji so navadno različne kombinacije točk, vektorjev in geometrijskih polj z atributnimi podatki. Morda je najbolj znani GIS te vrste Googlova spletna storitev Google Maps oz. njen namizni ekvivalent Google Earth. Z vnosom točk, ki jih opremijo z lastnimi opisi in slikami preko teh dveh storitev, mnogi uporabniki nevede sodelujejo pri uporabi in širjenju GIS-a.

Profesionalno programsko opremo GIS-a po drugi strani uporabljajo dejansko vsa velika podjetja v vseh industrijskih panogah. Zelo značilen pri-

mer so tudi javne službe, kot so komunalna podjetja, občinske in geodetske uprave. Te za shranjevanje podatkov največkrat uporabljajo baze podatkov kot so Oracle Spatial, Microsoft SQL Server, za prikaz in urejanje podatkov pa sisteme kot so ESRI-jev ArcGIS, MapInfo podjetja Pitney Bowes Software, MapGuide Open Source ali Autodeskov Infrastructure Map Server.

Večino podatkov, ki jih urejajo in hranijo komunalna podjetja ali geodetska uprava, potrebujejo pri obdelavi svojih projektov tudi gradbeni inženirji, vendar za njihovo delo sami programi GIS niso najprimernejša orodja, saj ne omogočajo načrtovanja inženirskih objektov. Zato projektanti potrebujejo hibridno orodje, ki omogoča branje podatkov iz sistemov GIS oziroma baz podatkov, slednji pa morajo imeti možnost izvoza podatkov v čim bolj obvladljive oblike. Na tem mestu je potrebno omeniti Autodeskov AutoCAD Map 3D. Gre za t. i. vertikalni proizvod v AutoCADovi veji, kar v osnovi pomeni, da je to AutoCAD s specifičnim naborom funkcij, ki segajo nad samo platformo. Naslednji v tej konkretni podveji je program AutoCAD Civil 3D, ki združuje vse funkcije AutoCAD Map 3D in jim dodaja še funkcije, specifične za področje nizkih gradenj.

AutoCAD Map 3D je proizvod, ki omogoča obdelavo podatkov GIS preko »Feature Data Object« ali krajše: tehnologije FDO. Poleg tega je v programu izjemno število dodanih orodij, ki omogočajo vnos geokodiranih rastrskih podlog, konverzijo koordinatnih sistemov, prikaz podatkov CAD na slojih iz spletnega servisa Bing Maps, izdelavo industrijskih modelov za prikazane objekte CAD (v osnovi to pomeni predhodno definirane načine prikaza različnih tipov infrastrukture, ki nato izboljšujejo možnosti analize te infrastrukture) ter uvoz, obdelavo in izvoz podatkov v pravzaprav katero koli obliko zapisa GIS, ki je trenutno na voljo, vključno z neposrednim priklopom na baze podatkov (z že omenjenim FDO). S tem je AutoCAD Map 3D postal odlično orodje v rokah prostorskih planerjev in nadzornikov večjih omrežij (kanalizacijski sistemi, električna omrežja, telekomunikacijska omrežja), hkrati pa ohranja uporabno vrednost za gradbene inženirje na področju nizkih in visokih gradenj.

Vseeno slednji v Sloveniji od celotnega orodja AutoCAD Map 3D še vedno uporabljajo le ozek nabor funkcij, ki je omejen na pregled slojev javne infrastrukture in parcelacije posameznih območij. Te podatke pridobivajo od Geodetske uprave Republike Slovenije, posameznih občinskih uradov in javnih komunalnih podjetij (kamor lahko dodamo še Telekom Slovenije in posamezne območne enote elektro podjetij). Osnovni namen pridobitve teh podatkov je pravilna umestitev arhitekturnih in gradbenih projektov v prostor ter pridobitev soglasij za izvedbo le-teh. Nabor uporabljenih funkcij je zato tako omejen, da je za projektante nakup licence programa AutoCAD Map 3D prevelik strošek.

Omenjeni podatki so pri različnih službah različno dostopni, saj nekateri še vedno ponujajo razmnožene karte z vrisanimi podatki. Toda na tem področju se dogajajo pozitivni premiki k tehnološki sodobnosti, kjer rastrske karte dobivajo vsaj podatek o geolokaciji, v večini primerov pa se je, vsaj v Republiki Sloveniji, uveljavil en tip zapisa vektorskih podatkov – to je »ESRI Shapefile« ali kratko »shapefile«. Za projektante s v tem formatu na voljo pravzaprav vsi podatki o parcelah v Republiki Sloveniji, prikazi rabe tal, gospodarska javna infrastruktura (GJI) in podobni sloji podatkov.

# Poglavje 4

# Uporabljene tehnologije in orodja

## 4.1 ESRI Shapefile

Predstavlja svetovno najbolj razširjen standard zapisa prostorskih vektorskih podatkov. Standard so razvili in predstavili v institutu ESRI že v začetku devetdesetih let prejšnjega stoletja. V datoteko shapefile (SHP) lahko zapišemo prostorske vektorske podatke za točke, linije in poligone. Običajno vsak element vsebuje še atributne podatke. Zaradi omejitev samega formata lahko datoteke shapefile vsebujejo le eno vrsto določenih prostorskih podatkov. V praksi to pomeni, da za popoln opis stanja neke infrastrukture od ustrezne službe dobimo tri nabore datotek SHP, kjer vsak nabor vsebuje ločene opise po točkah (npr. kanalizacijski jaški), linijah (npr. kanalizacijske cevi) in poligonih (npr. zadrževalni bazeni za meteorne vode ob avtocestah).

Standard ene datoteke shapefile pravzaprav sestavljajo tri obvezne datoteke: glavna datoteka ".shp", ki vsebuje geografske podatke; datoteka ".shx", ki vsebuje pozicijski indeks geometrije, kar omogoča hitro iskanje naprej in nazaj; ter datoteka ".dbf", kjer so atributni podatki objektov shranjeni v formatu dBASE, ki je eden izmed prvih formatov shranjevanja strukturiranih podatkov. Poleg teh glavnih datotek obstaja še cel nabor dodatnih datotek,

ki še dodatno določajo posamezne objekte, čeprav se v praksi ne uporabljajo pogosto [19].

## 4.2 OSGeo FDO

FDO (Feature Data Objects) je tehnologija, ki nam omogoča dostop do objektov (angl. feature) in manipulacijo z njimi na več različnih tipih izvora teh objektov. Z besedo "feature"v GIS-u označujemo objekte, ki združujejo podatek o geografski lokaciji objekta in poljubne atributne podatke oziroma lastnosti tega objekta.

Razvoj tehnologije so pričeli v podjetju Autodesk, kjer so potrebovali skupen mehanizem, ki bi jim omogočal dostop do več različnih izvorov prostorskih podatkov. Izvor teh podatkov predstavljajo različne geoprostorske (angl. geospatial) podatkovne baze in različni formati datotek (datoteke SHP, datoteke SDF itd.). Mehanizem je moral biti dovolj prilagodljiv, da je lahko zadovoljil potrebe različnih aplikacij.

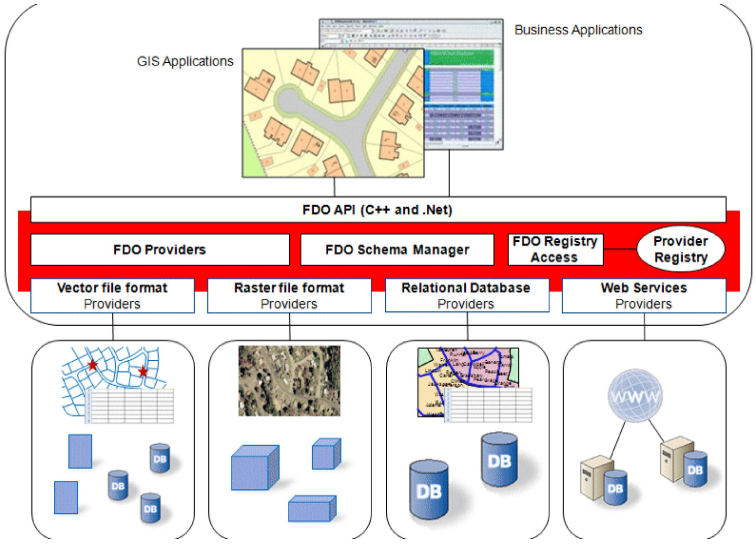
Prva verzija FDO-ja se je pojavila v programu Autodesk Map 3D 2005 in je omogočala dostop do podatkovne baze Oracle in datoteke SDF. Kasnejše verzije so dodale podporo še za MySQL, SQL Server, SHP, Raster, WFS, WMS idr. Autodesk je nato tehnologijo predal v odprtokodne vode z namenom, da bi spodbudil in pohitril razvoj ter razširil seznam izvorov podatkov, do katerih se lahko dostopa, kar bi razširilo uporabo te tehnologije [16].

### 4.2.1 FDO Data Access Technology

Z besedami FDO Data Access Technology označimo programski vmesnik, ki omogoča branje, urejanje, shranjevanje in analiziranje podatkov GIS-a na več različnih tipih izvora (MySQL, Oracle, SDF, SHP, WMS,...).

Tehnologija deluje po principu tako imenovanih ponudnikov (angl. provider), ki predstavljajo specifično implementacijo API-ja za določeni podatkovni izvor oziroma podatkovno shrambo (angl. data store). Nabor funkcio-

nalnosti API-ja, ki jih lahko določeni ponudnik zajame, je omejen z zmogljivostjo tehnologije podatkovnega izvora podatkov.




Slika 4.1: Princip delovanja FDO [16].

Aplikacije naprej vzpostavijo povezavo do določenega ponudnika, kjer se ustvari objekt, ki vsebuje povezavo (angl. connection object) med ponudnikom in podatkovno shrambo. Na podlagi tega objekta se nato dodatno ustvarijo še objekti, ki izvršujejo določene ukaze, kot so npr. SELECT, UPDATE, DELETE ipd., za samo podatkovno shrambo in se imenujejo ukazni objekti (angl. command objects). Pri izvajanju določenih ukazov, kot je na primer SELECT, ti ukazni objekti vračajo svoje objekte, ki vsebujejo rezultate izvedenega ukaza (angl. reader object). Aplikacije morajo same poskrbeti, da se objekti z rezultati pravilno prikažejo.

Vsako shrambo objektnih podatkov FDO, lahko predstavimo z naslednjo hierarhijo:

- Podatkovni izvor (angl. data source) vsebuje eno ali več podatkovnih shramb.
- Podatkovna shramba (angl. data store) vsebuje eno ali več objektnih

shem (angl. feature schema) in eno ali več prostorskih definicij (angl. spatial contexts).

- Objektna shema vsebuje enega ali več objektnih razredov (angl. feature class).
- Objektni razred, ki predstavlja objekt (angl. feature), vsebuje geometrijski podatek objekta, ter lahko še vsebuje en ali več atributnih podatkov objekta [16].

**Prostorska definicija** (angl. spatial contexts) zajema definicijo koordinatnega sistema, sferične parametre, merske enote, prostorske razsežnosti itd. za zbirko geometrij, ki pripadajo objektom.

**Objektna shema** (angl. schema) je poenostavljen in formaliziran opis tipa podatkov, ki se nahajajo v podatkovni shrambi. Predstavlja logičen opis tipov podatkov, ki se uporabljajo za ponazoritev predmetov v realnem svetu (električni vodi, električne postaje, kanalizacija, vodovod, jaški itd.) Sheme so metapodatki - podatki o podatkih. Zapisane so lahko v datotekah XML oziroma v posebnem podformatu GML, ki temelji na geografskem označevalnem jeziku (angl. Geographic Markup Language).

**Objektni razred** (angl. feature class) je element objektne sheme, ki opisuje tip predmeta, ki se nahaja v svetu. Sestavlja ga ime in definicije atributov, med katerimi so lahko tudi geometrijski atributi. Opisuje vse tipe podatkov, ki sestavljajo določeni objekt. Primer objektnega razreda je tabela v podatkovni bazi Oracle.

**Objekt** (angl. feature) predstavlja abstrakcijo naravnega ali umetno narejenega predmeta, ki se nahaja v svetu. Vsak objekt je neposredno ali posredno povezan z določeno geografsko lokacijo. Objekti so prostorski objekti ali neprostorski objekti. Prostorski objekti vsebujejo vsaj eno geometrijsko lastnost, npr. jaški so predstavljeni s točkami, vodovodi z linijami in parcele s poligoni. Neprostorski objekti ne vsebujejo geometrijskih lastnosti, vendar so lahko povezani s prostorskimi objekti, npr. ventil, ki se nahaja na vodovodni liniji [5].

### 4.2.2 FDO Provider for SHP

Ponudnik (angl. provider) FDO za SHP omogoča branje in pisanje prostorskih in atributnih podatkov iz podatkovne strukture ESRI SHP in v njo. To strukturo sestavljajo že prej omenjene datoteke SHP (geometrija), SHX (indeksi) in DBF (atributi v formatu dBASE). Ponudnik FDO dostopa do podatkov v vseh treh datotekah. Vsako datoteko SHP ter njeno pripadajočo datoteko DBF, obravnava kot en objektni razred, ki vsebuje podatek o geometriji in dodatne atributne podatke. Ponudnik FDO za datoteke SHP se lahko uporablja v operacijskih sistemih Windows ali Linux [6].

## 4.3 Qt

Qt je aplikacijsko ogrodje, ki se uporablja za razvoj aplikacij s poudarkom na grafičnem uporabniškem vmesniku, razvoj konzol za strežnike in razvoj orodij ukazne vrstice. Qt poleg standardnega jezika C++ uporablja še posebni generator kode, imenovan "Meta Object Compiler", ki skupaj z določenimi makri obogati celotni jezik. Deluje v vseh glavnih računalniških okoljih in hkrati tudi v nekaterih mobilnih okoljih. Med znane aplikacije, ki so v celoti ali vsaj določeni meri razvite z uporabo Qt-ja, štejemo: VLC, Google Earth, Opera, TortoiseHg, Texmaker, Skype, Autodesk Maya, ... [17].

Qt/MFC Migration Framework je ogrodje, ki omogoča migracijo obstoječih aplikacij Win32 ali MFC na ogrodje Qt-ja in obratno. Ogrodje ponuja seznam razredov, ki omogočajo uporabo Qt in oken Win32/MFC v isti aplikaciji. Hkrati omogoča, da se že obstoječi elementi, razviti s uporabo Win32 ali MFC, vgradijo v Qt-jeve dodatne priročne okenske gradnike (angl. widgets) [18].

## 4.4 Visual Studio 2010

Microsoft Visual Studio je najbolj razširjeno razvojno okolje, za razvoj spletnih aplikacij ASP .NET, spletnih storitev ter namiznih in mobilnih aplikacij.

Visual Studio vsebuje urejevalnik kode, ki podpira IntelliSense. Zelo uporaben del Visual Studio je razhroščevalnik (angl. debugger), s katerim se lažje odkrijejo napake v kodi. Poleg že naštetih orodij vključuje še orodja za gradnjo aplikacij z grafičnim uporabniškim vmesnikom, grajenje spletnih strani in orodja za delo s podatkovnimi bazami.

Podpira različne programske jezike, kot so C, C++, C#, Visual Basic. Poleg Microsoftovih programskih jezikov podpira še Python, Ruby, HTML, CSS, JavaScript, ... [21].

### 4.4.1 Visual C++

Microsoft Visual C++ je integrirano/interaktivno razvojno okolje za programska jezika C in C++. Uporablja se za razvoj aplikacij, ki temeljijo na Microsoft Windows API-ju in Microsoft .NET Frameworku. Poleg razvoja standardnih aplikacij z grafičnim uporabniškim vmesnikom Visual C++ omogoča razvijalcem razvoj spletnih aplikacij, aplikacij tipa pametni odjemalec (angl. smart-client), ki temeljijo na Windows API-ju, ter razvoj aplikacij, tipa tanki odjemalec (angl. thin-client) in pametni odjemalec, za mobilne naprave [22].

Veliko aplikacij, razvitih z uporabo okolja Visual C++, za pravilno delovanje potrebuje tako imenovane distributivne pakete Visual C++. Ti paketi predstavljajo standardne knjižnice, ki se uporabljajo pri razvoju aplikacij. Več aplikacij, ki uporabljajo iste knjižnice, lahko hkrati dostopa do teh paketov, kar omogoča, da je v sistemu potrebna samo ena kopija določenega paketa.

## 4.5 HTML

HTML (HyperText Markup Language) je označevalni jezik za izdelavo spletnih strani in ostalih podatkov, ki se lahko prikažejo v spletnem brskalniku. HTML dokument je v osnovi tekstovna datoteka, ki vsebuje element HTML. Elementi HTML so značke (angl. tags), ki se nahajajo znotraj znakov '<', '>' in predstavljajo osnovne gradnike vseh spletnih strani [9].

## 4.6 CSS

CSS (Cascading Style Sheets) je jezik, ki omogoča stilizacijo prikaza dokumentov, napisanih v označevalnem jeziku (angl. markup language). Najpogosteje se uporablja za oblikovanje sloga spletnih strani napisanih v jeziku HTML in XHTML, vendar se lahko uporablja tudi v dokumentih XML. Glavni namen razvoja jezika je bila ločitev vsebine in sloga dokumenta, kar poveča preglednost in izboljša urejanje celotne kode.

Sestavljajo ga pravila, s katerimi določimo oblikovne lastnosti elementov, kot so barve, velikosti, odmike, poravnave, obrobe, pozicije in vrsto drugih atributov. Dovoljuje, da en element uporablja več pravil hkrati ter da isto pravilo lahko uporablja več elementov hkrati [3].

## 4.7 JavaScript

JavaScript je visoko nivojski, dinamični, tolmačeni programski jezik (angl. interpreted programming language), ki omogoča razvoj dinamičnih spletnih strani. JavaScript je vgrajen v gostiteljsko okolje (angl. host environment), ki je običajno spletni brskalnik [4].

Podpira tako objektno usmerjeni kot funkcijski stil programiranja. Njegova sintaksa izvira iz Jave, prvorazredne (angl. first-class) funkcije iz programskega jezika Scheme in prototipnega dedovanja (angl. prototype-based inheritance) iz programskega jezika Self [4]. Kljub podobnosti med imenoma Java in JavaScript gre za povsem različna programska jezika, ki ju povezuje samo izvor sintakse JavaScript.

Programski jezik JavaScript, ki je bil razvit v podjetju Netscape, štejemo med trojico tehnologij, ki jih mora znati vsak spletni razvijalec: HTML, s katerim določimo vsebino spletne strani; CSS, s katerim določimo stil spletne strani; in JavaScript, s katerim določimo obnašanje spletne strani [4].

### 4.7.1 JQuery

JQuery je knjižnica JavaScript, ki na enostaven način omogoča manipulacijo dokumentov HTML, izbiro elementov DOM, izdelavo animacij, razvoj aplikacij AJAX idr. JQuery je najbolj razširjena knjižnica JavaScript na svetu [10].

Knjižnica ima API, ki je zelo preprost za uporabo in deluje na vseh bolj znanih spletnih brskalnikih, ter omogoča enostavno pisanje gradnikov (angl. widgets), ki dodatno razširijo funkcionalnost knjižnice.

### 4.7.2 JQuery UI

JQuery UI je knjižnica JavaScript zgrajena na podlagi knjižnice JQuery. Združuje sklop gradnikov (angl. widgets), interakcij uporabniškega vmesnika (angl. user interface interactions), učinkov (angl. effects) in stilov (angl. themes), ki se uporabljajo za gradnjo visoko interaktivnih spletnih aplikacij [11].

Pri razvoju aplikacije, ki vsebuje knjižnico jQuery UI, je v razvoj nujno potrebno vključiti še knjižnico jQuery.

## 4.8 NSIS

NSIS (Nullsoft Scriptable Install System) je skriptni instalacijski sistem za ustvarjanje namestitvenih programov za operacijski sistem Windows, razvit v podjetju NullSoft. Njegov prvotni namen je bila distribucija programa Winamp. Podjetje NullSoft se je po določeni interni iteraciji razvoja odločilo, da se celotni projekt postavi na spletni kodni repozitorij SourceForge. S tem je sistem NSIS postal odprto kodni projekt, kar je povzročilo močno razširitev njegovega razvoja in uporabe [13].

NSIS omogoča izdelavo namestitvenih programov, sposobnih izvajanja namestitve, odstranitve, upravljanja s sistemskim registrom, ekstrakcije datotek ipd. Podpira vse glavne različice operacijskega sistema Windows, na

voljo ima tri različne metode kompresije podatkov (ZLib, BZip2, LZMA), podpira več jezikov uporabniškega vmesnika v enem namestitvenem programu in izdelavo poljubnih uporabniških vmesnikov. NSIS lahko hkrati komunicira z naborom vtičnikov, ki dodatno razširijo delovanje sistema in med drugim omogočajo izdelavo posodobitev obstoječih datotek, povezovanje z internetom, prenos datotek iz interneta itd. [14].

## 4.9 Mercurial version control

Mercurial je odprtokodno orodje za nadzor izvorne kode. Za razliko od tradicionalnih sistemov za nadzor izvorne kode, ki jih običajno sestavlja arhitektura odjemalec-strežnik (angl. client-server), Mercurial predstavlja porazdeljen sistem. Porazdeljen sistem nadzora pomeni, da ima vsak razvijalec lokalno kopijo celotne zgodovine razvoja, ker omogoča razvoj, neodvisen od centralnega strežnika in internetne povezave.

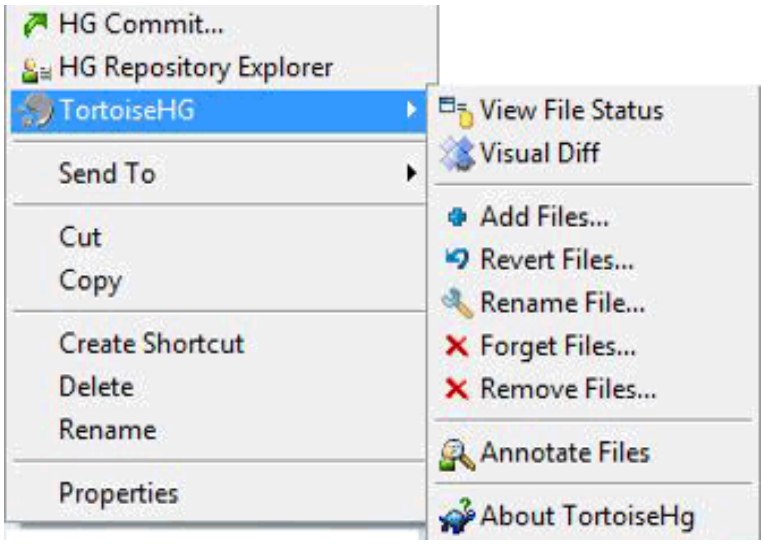
Mercurial je skoraj v celoti napisan v programskem jeziku Python in vključuje primerjalnik binarnih datotek, ki je zaradi boljših zmogljivosti napisan v programskem jeziku C. Deluje v vseh operacijskih sistemih, kot so Windows, Mac OS X in Linux.

Osnovne funkcije Mercuriala uporabniku med drugim omogočajo generiranje različic (angl. diffs) med določenimi verzijami izvorne kode, vračanje v zgodovino razvoja, ustvarjanje oznak (angl. tags) sklopa sprememb kode ipd.. Osnovne funkcije se lahko dodatno razširijo z uporabo dodatkov (angl. extensions). Ti dodatki so napisani v programskem jeziku Python in lahko spreminjajo delovanje osnovnih ukazov, dodajajo nove ukaze in imajo dostop do vseh osnovnih funkcij [12].

### 4.9.1 TortoiseHg

TortoiseHg je nabor grafičnih orodij in razširjena ukazna lupina (angl. shell extension) za orodje Mercurial. Gre za odprtokodni projekt, ki je v večini napisan v programskih jezikih Python in PyQt. TortoiseHg, v operacijskih

sistemih Windows sestavlja razširjena ukazna lupina, ki v datotečni brskalnik doda prekrivne ikone in dodatne vsebinske menije za delo z orodjem Mercurial [20].




Slika 4.2: Vsebinski meni orodja TortoiseHg.

# Poglavje 5

# Aplikacija - SHPReader

## 5.1 Namen aplikacije

Poglavitno težavo, ki jo želimo z diplomsko nalogo izpostaviti in premostiti, je pomanjkanje povezanosti med cenovno ugodnejšimi alternativni programi CAD in sistemi GIS. Ravno z namenom premostitve te težave se v diplomski nalogi osredotočimo na popolnoma praktičen način izvedbe funkcionalnosti, ki poveže program CAD s podatki iz sistema GIS. Cilj diplomske naloge je v celoti razviti aplikacijo, ki uporabnikom, predvsem projektantom, omogoča prikazovanje podatkov iz datotek SHP v programih CAD, natančneje v programu BricsCAD.

Osredotočenje na datoteke SHP je pogojeno z dejstvom, da so to najbolj razširjene oziroma najbolj priljubljene oblike zapisa podatkov GIS, ki jih projektantom ponujajo komunalna podjetja in Geodetska uprava Republike Slovenije. Razvoj aplikacije za program BricsCAD smo izvedli zaradi dejstva, da je dostop do SDK-ja nekoliko enostavnejši kot pri ZwCAD+. Tudi odzivnost in želja po sodelovanju je bila s strani podjetja Bricsys pravzaprav takojšnja. Dodatni razlog za razvoj aplikacije za ta program je tudi dejstvo, da se BricsCAD razvija v smeri, ki ga vizualno loči od AutoCAD-a, in se s tem oddaljuje od negativne oznake, da je klon drugega proizvoda.

## 5.2 Razvoj aplikacije

### 5.2.1 Analiza in načrtovanje

Razvoj aplikacije smo začeli z analizo uporabniških zahtev in načrtovanjem izvedbe same aplikacije, saj je takšna začetna zastavitev vsakega razvoja hkrati tudi najbolj ključen korak ali stopnja v procesu razvoja programske opreme.

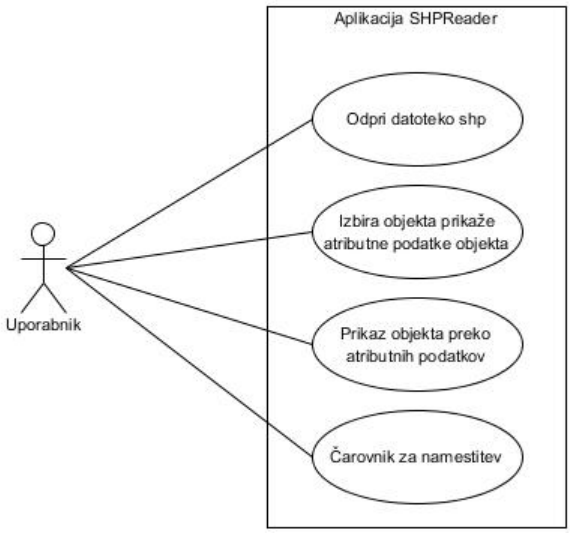
Analiza uporabniških zahtev in potreb nam omogoča določiti vse funkcionalnosti aplikacije, prepreči možnost pojavljanja dodatnih zahtev ter povsem odstrani ali vsaj v ogromni meri zmanjša kasnejše korekcije osnovnih funkcionalnosti, pridobljene iz uporabniških zahtev. Natančna in temeljita izvedba te faze razvoja nam omogoča lažjo in predvsem pravilno implementacijo svoje aplikacije. V to fazo se lahko vključuje tudi naročnik aplikacije, zaradi česar je potrebno tesno sodelovanje med naročnikom in razvijalcem, da se lahko določijo ustrezne zahteve aplikacije in da se izognemo kasnejšim nesoglasjem. Pri razvoja svoje aplikacije smo se naslonili na nasvet in mnenje izkušenega projektanta nizkih gradenj pri podjetju Proarc d.o.o., gospoda Andreja Sedeja, univ. dipl. inž. grad., ki se z vprašanji vnosa prostorskih podatkov v gradbene projekte srečuje vsakodnevno. S pomočjo pridobljenih informacij smo lahko predvideli potrebe in zahteve potencialnih uporabnikov aplikacije.

Opredelili smo naslednje zahteve:

- Aplikacija se mora preprosto vključiti v program BricsCAD. Aplikacija naj se uporabniku prikaže z uporabo enega ukaza.
- Aplikacija naj omogoča preprost način izbire datotek .shp, ki jih želimo videti v risbi/trenutno odprti datoteki .dwg.
- Aplikacija mora omogočiti prikazovanje oziroma omogočiti dodajanje objektov iz datoteke .shp v risbo/trenutno odprto datoteko .dwg. To predstavlja prvo polovico osnovne funkcionalnosti aplikacije.
- Aplikacija mora omogočati prikazovanje atributnih podatkov iz datoteke shp, kar je druga polovica osnovne funkcionalnosti aplikacije.

- Uporabniku naj se prikažejo atributni podatki dodanega objekta, če tega izbere na risbi.
- Uporabnik lahko s seznama atributnih podatkov izbere in nato približa objekt, ki mu pripadajo določeni atributni podatki.
- Aplikacija naj omogoča prikazovanje več datotek .shp naenkrat.
- Aplikacija naj vsako datoteko .shp doda na svoj sloj v risbi. To uporabniku omogoča filtriranje prikaza podatkov.
- Uporabniku prijazen in intuitiven uporabniški vmesnik.
- Uporabniku omogočiti preprosto namestitev in odstranitev aplikacije.

Za dodatno razumevanje funkcionalnih zahtev smo izdelali še diagram primera uporabe, ki predstavlja vizualen prikaz komunikacije uporabnika in aplikacije oziroma sistema.




Slika 5.1: Diagram primera uporabe.

V fazi načrtovanja smo določili strukturo sistema na podlagi zahtev, pridobljenih pri analizi, ter se odločili za agilnejši pristop razvoja aplikacije,

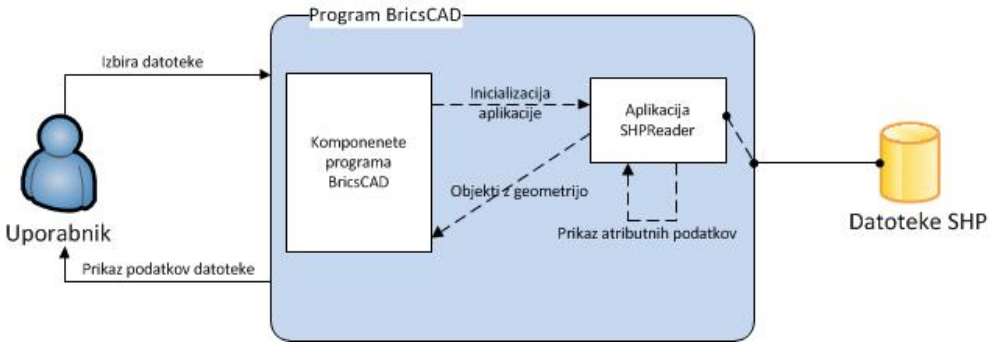
kjer smo dali večji poudarek delujoči kodi kot obsežni dokumentaciji. Med načrtovanjem smo tudi raziskali možnost uporabe tujih knjižnic z namenom hitrejšega razvoja aplikacije ter umaknili zahtevo po razvoju kompleksnih funkcij, za katere že obstajajo preverjene in pogosto uporabljene alternative.

Za vzpostavitev povezave med svojo aplikacijo in datotekami shp smo se odločili za uporabo knjižnice FDO, ki nam omogoča preprost dostop do objektov in njihovih podatkov v okviru različnih tipov izvora. Hkrati nam uporaba te knjižnice zagotavlja lažjo nadgradnjo dostopa naše aplikacije do drugih tipov izvora podatkov.

Za razvoj logike aplikacije smo se zaradi robustnejšega in potencialno hitrejšega delovanja aplikacije odločili uporabiti programski jezik C++, kar pa je po drugi strani predstavljalo težavnejšo izgradnjo enostavnega in učinkovitega uporabniškega vmesnika. Iz istega razloga smo se odločili, da pri izdelavi uporabniškega vmesnika ne uporabimo izključno knjižnice MFC. Za boljšo možnost se je izkazala uporaba ogrodja Qt, ki omogoča hitro in preprosto gradnjo grafičnih uporabniških vmesnikov. Ogrodje Qt omogoča uporabo programskega jezika JavaScript in drugih elementov spletnega razvoja (HTML, CSS,...) v isti aplikaciji, ki je bila razvita v programskem jeziku C++. To se je pokazala kot kombinacija, ki jo je bilo vredno uporabiti, in s tem preveriti, kako poteka komunikacija med elementi, napisanimi v različnih programskih jezikih. Logiko aplikacije smo tako napisali v programskem jeziku C++, uporabniški vmesnik pa v jezikih spletnega programiranja (JavaScript, HTML, CSS ...).

Program BricsCAD v okolju Windows za prikaz uporabniškega vmesnika uporablja elemente knjižnice MFC, ki se jim tako nismo mogli v celoti izogniti. Poleg ogrodja Qt smo zato uporabili še dodatek - knjižnico Qt/MFC Migration Framework. Ta vsebuje razrede, ki omogočajo, da elemente Qt-ja vpnemo v elemente MFC-ja.

Kot smo že omenili, je aplikacija, ki smo jo razvili v tej diplomski nalogi, vpeta v večji program BricsCAD, in pod nobenim pogojem ne more delovati samostojno. Zato aplikacijo SHPreader opredelimo kot nadgradnjo, oziroma vtičnik programa BricsCAD. Uporabnik bo pri uvozu datoteke shp navidezno uporabljal samo program BricsCAD, kljub temu pa slednji pri celotni operaciji sodeluje le z vizualnim prikazom objektov. Aplikacija SHPReader opravi vse glavne naloge uvoza datoteke shp v program BricsCAD. Za boljše razumevanje celotnega sistema smo dodali še diagram arhitekture le-tega.




Slika 5.2: Arhitektura celotnega sistema.

### 5.2.2 Opis razvoja in delovanja aplikacije

Razvoj aplikacije smo naslonili na ugotovitve, pridobljene v fazah analize in načrtovanja. Funkcionalnosti smo razvijali postopoma, po korakih, zastavljenih v fazi analize. Razvoj je potekal v iteracijah razvijanja in testiranja, tako da smo naprej razvili določeno funkcionalnost, ter jo nato v celoti testirali in odpravili napake, preden smo se lotili naslednje uporabniške zahteve določene, v fazi analize.

Naprej smo omogočili klic aplikacije iz programa BricsCAD. To je hkrati tudi začetni korak pri razvijanju aplikacij za programe, kot so BricsCAD, ZwCAD in AutoCAD. V ta namen smo vpeljali funkcijo vstopne točke (angl. entry point function), ki poskrbi, da naša aplikacija in program BricsCAD lahko med seboj komunicirata. Hkrati smo morali tukaj registrirati ukaz, s katerim bomo poklicali svojo aplikacijo. Spodaj je prikazan del izvorne kode funkcije vstopne točke, s katero smo poskrbeli za registracijo ukaza. Vsakemu ukazu, ki ga registriramo in dodamo, moramo obvezno dodati tudi funkcijo, ki jo ukaz kliče.

```
//registracija ukaza
ACED_ARXCOMMAND_ENTRY_AUTO(CSHPReaderApp, Brx, SHPReader,
SHPReader, ACRX_CMD_TRANSPARENT, NULL)

//ukaz SHPReader ::: moramo dodati funkcijo, ki se klice
//v programu BricsCAD, kadar uporabimo ukaz "SHPReader"
static void BrxSHPReader(void){
  //inicializacija/prikaz osnovnega okna aplikacije
  cmdShowPalette();
}
```

Po uspešni integraciji svoje aplikacije v program BricsCAD smo želeli vzpostaviti komunikacijo med logiko svoje aplikacije (programski jezik C++) in grafičnim uporabniškim vmesnikom (programskim jezikom JavaScript, HTML, CSS ...). V inicializaciji aplikacije SHPReader v svoje okno dodamo gradnik QWinWidget, za kar poskrbi funkcija cmdShowPalette(). Gra-

dnik QWinWidget je del knjižnice Qt/MFC Migration Framework in nam omogoča souporabo elementov knjižnice Qt z elementi uporabniškega vmesnika, ki sloni na knjižnici MFC tako kot uporabniški vmesnik programa BricsCAD. V gradnik QWinWidget smo nato vpeli še gradnik QWebView, s katerim prikazujemo in urejamo spletne dokumente.

Delovanje ogrodja Qt sloni na principu signalov in rež (angl. signals and slots). Kadar se v elementih Qt-ja zgodi neki dogodek, le-ta proži tako imenovani signal. Reže pa predstavljajo funkcije, ki se kličejo kot odgovor na ta signal. Osnovne razrede ogrodja Qt lahko razširimo v poljubne razrede, ki jim dodamo lastne signale in reže. To smo tudi storili in s tem vzpostavili povezavo med programskim jezikom C++ in programskim jezikom Javscript. Naprej smo razširili osnovni razred "QObject" v razred "c_QtFormExtractor" in mu dodali naslednjo metodo:

```
void c_QtFormExtractor::attachObject(){
  frame->addToJavaScriptWindowObject( QString("MyApi"), this );
}
```

V tej metodi z uporabo funkcije "addToJavaScriptWindowObject" svoj razred in vse njegove javne reže oziroma metode izpostavimo programskemu jeziku Javascript pod izbranim imenom "MyApi". Naš razred se kot podrejeni element doda osnovnemu elementu spletne strani. Klic funkcije C++ iz programskega jezika Javascript nato opravimo po naslednjem principu:

```
function selectSHPFile(){
  MyApi.selectFile_JS();
}
```

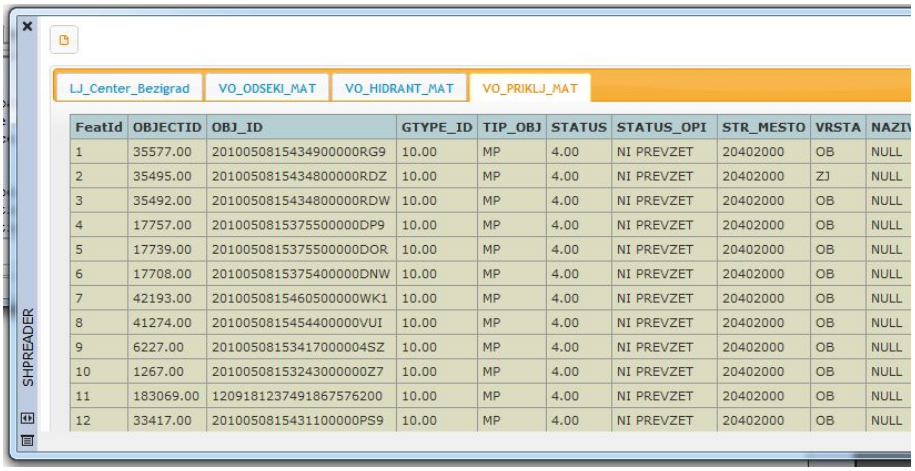
Prej omenjeno metodo "attachObject" smo morali dodati še v samo inicializacijo svojega gradnika QWebView, kar storimo v naslednji metodi, ki je del širše funkcije:

```
void c_QtFormExtractor::setWebView( QWebView *view ){
  QWebPage *page = view->page();
  frame = page->mainFrame();

  attachObject();
  connect( frame, SIGNAL(javaScriptWindowObjectCleared()),
  this, SLOT(attachObject()) );
}
```

Tukaj bi izpostavili še uporabo funkcije "connect", ki nam ohranja povezavo med programskima jezikoma v primeru, da v uporabniškem vmesniku odpremo oziroma preidemo na novo spletno stran. Kadar prehajamo med spletnimi stranmi se sprosti element "document object", ki predstavlja celotno spletno stran in je osnovni gradnik vseh spletnih strani HTML. Povezavo pa izgubimo, ker je naš razred, kot smo že omenili, v bistvu dodan kot podrejeni element tega glavnega elementa. Samo delovanje ogrodja Qt nam tukaj ponudi rešitev iz te zagate. Ob sprostitvi glavnega elementa spletne strani se sproži signal "javaScriptWindowObjectCleared". Mi smo na ta signal z uporabo funkcije "connect" dodali lastno režo, metodo "attachObject", in tako poskrbeli za vzpostavljanje povezave med programskima jezikoma tudi v primeru nalaganja različnih strani oziroma različnih komponent svojega uporabniškega vmesnika.

S tem smo si pripravili vse potrebne pogoje za začetek razvoja uporabniškega vmesnika. Naprej smo pripravili preprosto spletno stran, kjer se nahaja gumb, s katerim smo uporabniku omogočili izbor datotek shp. Določili smo mesto, kamor lahko vpnemo atributne podatke, ter pripravili začetne funkcije, ki jih programska jezika C++ in JavaScirpt uporabljata za medsebojno komunikacijo.

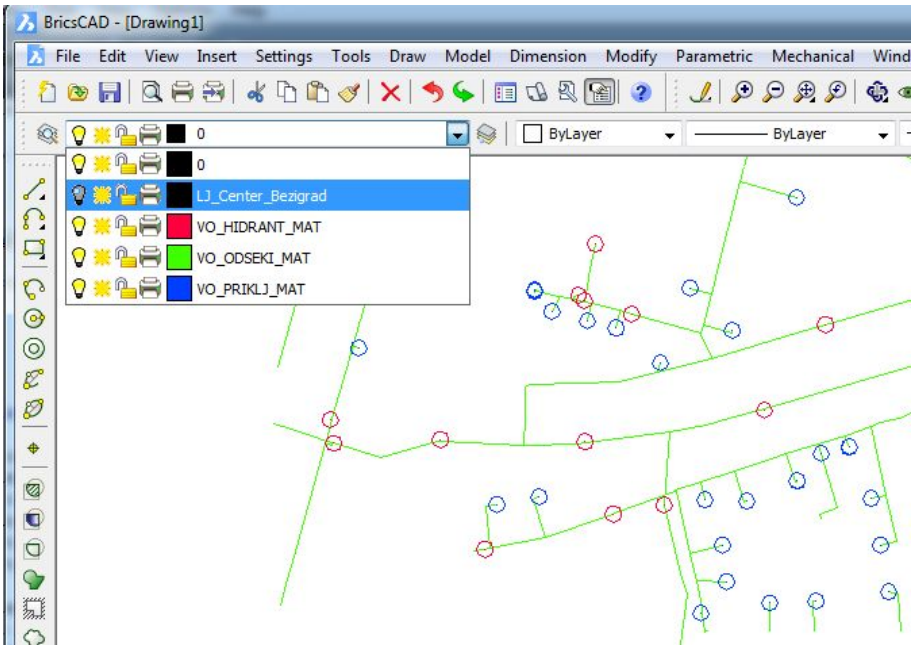



| FeatId | OBJECTID | OBJ_ID | GTYPE_ID | TIP_OBJ | STATUS | STATUS_OPI | STR_MESTO | VRSTA | NAZI |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 35577.00 | 2010050815434900000RG9 | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | OB | NULL |
| 2 | 35495.00 | 2010050815434800000RDZ | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | ZJ | NULL |
| 3 | 35492.00 | 2010050815434800000RDW | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | OB | NULL |
| 4 | 17757.00 | 2010050815375500000DP9 | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | OB | NULL |
| 5 | 17739.00 | 2010050815375500000DOR | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | OB | NULL |
| 6 | 17708.00 | 2010050815375400000DNW | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | OB | NULL |
| 7 | 42193.00 | 2010050815460500000WK1 | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | OB | NULL |
| 8 | 41274.00 | 2010050815454400000VUI | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | OB | NULL |
| 9 | 6227.00 | 201005081534170000O4SZ | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | OB | NULL |
| 10 | 1267.00 | 2010050815324300000OZ7 | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | OB | NULL |
| 11 | 183069.00 | 1209181237491867576200 | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | OB | NULL |
| 12 | 33417.00 | 2010050815431100000PS9 | 10.00 | MP | 4.00 | NI PREVZET | 20402000 | OB | NULL |

Slika 5.3: Prikaz uporabniškega vmesnika naše aplikacije, potem ko smo uvozili datoteko shp.

S klikom na gumb za izbor datoteke iz jezika JavaScript v jezik C++ kličemo razred "CFileDialog", ki nam prikaže standardno Windowsovo okno za odpiranje in shranjevanje datotek. Po izboru datoteke naprej preverimo, ali je uporabnik predhodno že odprl to datoteko. Če je uporabnik datoteko že kdaj uvozil, ga o tem obvestimo in mu ponudimo možnost ponovnega uvoza ali prekinitev izvajanja procesa uvoza. Zatem vzpostavimo povezavo na datoteko SHP in preberemo podatke, ki se nahajajo v izbrani datoteki. Za izbrano datoteko pripravimo lasten sloj in ga dodamo v trenutno odprto risbo v programu BricsCAD. Iz pridobljenih podatkov izluščimo podatke o geometriji, kar vključuje tako koordinate kot tip geometrije - točko, linijo ali poligon. Iz geometrije sestavimo objekte in jih pripnemo na prej ustvarjeni sloj, atributne podatke teh objektov pa prikažemo v svoji spletni strani

oziroma uporabniškem vmesniku. Slika 5.4 prikazuje primer uporabe, kjer je uporabnik uspešno uvozil več datotek shp z različnimi tipi geometrije in poimenske sloje, ki mu jih je naša aplikacija dodala v program BricsCAD. Sloj z imenom "`LJ_Center_Bezigrad`" je v tem primeru izključen in zaradi tega tudi niso vidni objekti, ki smo mu jih dodali.

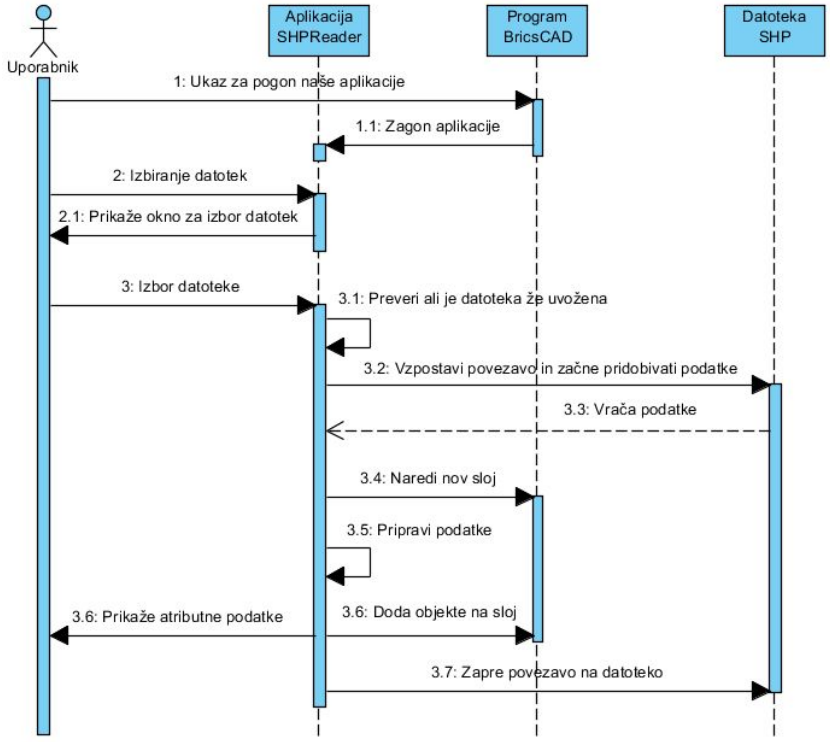



Slika 5.4: Prikaz dodanih slojev in objektov, ki jih je naša aplikacija uvozila iz različnih datotek.

Prikazovanje atributnih podatkov smo prvotno želeli v celoti implementirati v uporabniškem vmesniku, kar pomeni, da bi logika naše aplikacije samo posredovala atributne podatke vsakega objekta v vmesnik, kjer bi jih neposredno uredili in dodali v tabelo za prikaz. Na žalost se je pri večjih datotekah ta princip delovanja izkazal za zelo nestabilnega (usodne napake aplikacije) in počasnega (več minutno nalaganje vmesnika). Težavo smo rešili tako, da v programskem jeziku C++ sestavimo elemente tabele HTML, ki jo uporabljamo za prikaz atributnih podatkov. Za vsak objekt prebrane datoteke v jeziku C++ sestavimo celotno vrstico tabele, nato v programskem jeziku

JavaScript samo združimo vse elemente tabele v celoto in podatke prikažemo uporabniku.

Primer uporabe, kjer uporabnik izbere in odpre novo datoteko shp, smo za boljšo predstavo izvedli z uporabo sekvenčnega diagrama, ki je prikazan na Sliki 5.5. V sekvenčnem diagramu smo na osnovi časovnega zaporedja prikazali izmenjavo sporočil med objekti, ki sodelujejo v našem sistemu, za izbrani primer uporabe.




Slika 5.5: Diagram zaporedja uvoza datoteke shp v program BricsCAD z uporabo naše aplikacije.

Kot smo predhodno že omenili, smo za vzpostavitev povezave in pridobivanja podatkov iz datotek shp, uporabili odprtokodno knjižnico FDO. Za uporabo knjižnice smo se odločili, ker predstavlja odlično in mnogokrat uporabljeno rešitev za vzpostavitev povezave, pridobivanje in urejanje podatkov podatkovnih shramb GIS, s čimer smo se lahko izognili lastnemu razvoju

kompleksne logike za vzpostavitev teh procesov. Omenjeni razvoj tudi ni bil predmet te diplomske naloge in smo tako prihranili ogromno časa pri izdelavi svoje aplikacije.

Če si datoteko shp predstavljamo kot podatkovno shrambo, nam knjižnica omogoča izvajanje osnovnih operacij pridobivanja podatkov ("select"), urejanje podatkov ("update"), dodajanje podatkov ("insert") in brisanje podatkov ("delete") za to datoteko ali katero koli drugo vrsto izvora podatkov. V svoji aplikaciji smo implementirali enosmerno komunikacijo, kar pomeni, da za izbrano datoteko shp opravljamo operacije pridobivanja podatkov ("select"). Uporaba knjižnice FDO nam je narekovala tudi standardiziran postopek za pridobivanje podatkov iz izvora podatkov.

Preden smo lahko vzpostavili povezavo na datoteko, smo morali določiti tip povezave. Tega določimo glede na vrsto izvora podatkov. V svoji aplikaciji podpiramo samo datoteke shp. Tako smo lahko neposredno določili tip povezave brez dodatnega preverjanja izvora podatkov. Po nastavitvi tipa in uspešni vzpostavitvi povezave do datoteke smo se morali sprehoditi čez celotno hierarhijo shrambe podatkov (podrobnejši opis hierarhije se nahaja v 4. poglavju o uporabljenih tehnologijah in orodjih), da smo prišli do geometrijskih in atributnih podatkov objektov. Naprej smo pridobili seznam vseh shem, ki se nahajajo v naši podatkovni shrambi, opravili sprehod po seznamu in pridobili vse objektne razrede (angl. feature class).Nato smo pripravili iterator, ki se sprehodi po podatkih vseh objektnih razredov in nam omogoča izluščiti geometrijske in atributne podatke vseh objektov v datoteki.

Preostali sta nam še dve večji uporabniški zahtevi, ki smo ju uspeli zajeti v okviru same kode aplikacije. Prva uporabniška zahteva je označitev atributnih podatkov, kadar uporabnik izbere objekt na risbi. Druga uporabniška zahteva v tem sklopu je, da se uporabniku prikaže in označi objekt v risbi DWG, ko uporabnik izbere vrstico z atributnim podatkom tega objekta v tabeli atributnih podatkov.

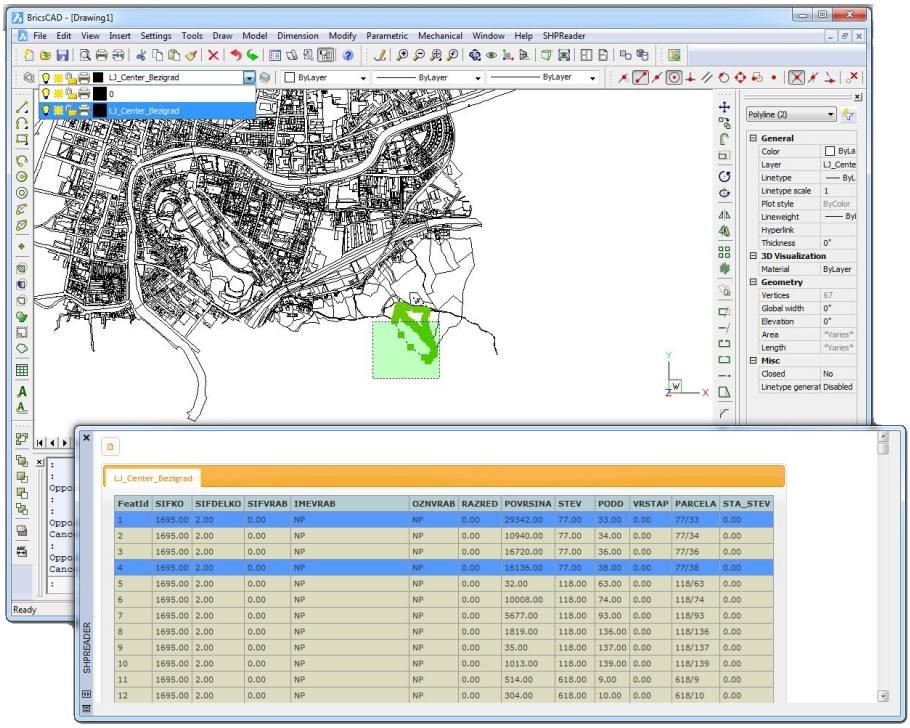
Za reševanje prve izmed teh dveh zahtev smo v svojo aplikacijo dodali še razred "c_EditorReactor", ki je izpeljan iz razreda "AcEditorReactor".

Ta nam omogoča spremljanje funkcij, ki jih uporablja urejevalnik programa BricsCAD. Urejevalnik je vpet v risalno območje programa BricsCAD in se na dogodke odziva s klicem funkcij, ki so dodeljene dogodkom. Z uporabo izpeljanega razreda "c_EditorReactor" smo lahko te osnovne funkcije povozili oziroma nadgradili z lastnimi funkcionalnosti. Kadar uporabnik izbere poljuben objekt na risbi, se sproži funkcija "pickfirstModified()". Tej funkciji dodamo lastne naloge. Iz izbranega objekta smo pridobili njegovo identifikacijsko številko. Unikatna identifikacijska številka je samodejno dodeljena vsakemu objektu, ki se doda na risbo. To unikatno številko smo že v fazi razvoja osnovnih funkcij za dodajanje objektov na risbo in funkcij za prikazovanje atributnih podatkov objektov zapisali v identifikacijski atribut vsake vrstice v tabeli atributnih podatkov. To smo storili zato, da smo ohranili povezanost med samimi objekti in njihovimi atributnimi podatki. Po pridobitvi unikatne identifikacijske številke objekta smo v svoji funkciji "pickfirstModified()" morali le še poiskati vrstico, ki pripada tem objektu in jo vizualno prikazati uporabniku. To storimo s klicem naslednjih funkcij uporabniškega vmesnika naše aplikacije:

```
//Funkcija uporabniku ''pribliza'' atributne podatke izbranega objekta
function scrollToSelectedObject(objectID) {
  var object = document.getElementById(objectID);
  if(object!=null){
    var tableParentTag = $(object).parent().parent();
    var tableIDvalue = $(tableParentTag).attr('id');
    var indexArray = tableIDvalue.split("_");
    $( "#fileTabs" ).tabs( "option", "active", indexArray[1]-1 );
    object.scrollIntoView(true);
  }
}
//Funkcija doda poseben stil vrstici izbranega objekta
function addSelectedStyle(objectID){
  var object = document.getElementById(objectID);
  if(object!=null)  $(object).addClass("selectedObject");
}
```

Za implementacijo druge zahteve v tem sklopu, ki ima pomensko obratno funkcijo kot ravnokar opisana zahteva, smo morali izbrati drugačen pristop. S tem, ko uporabnik izbere določeno vrstico v tabeli atributnih podatkov (dvojni klik na vrstico), naprej s funkcijo "addSelectedStyle" obarvamo izbrano vrstico. Uporabniški vmesnik nato posreduje identifikacijsko številko objekta, zapisano v identifikacijskem atributu vrstice, logiki naše aplikacije. V logiki ne glede na vrsto objekta (točka, linija ali poligon) pridobimo koordinate, ki predstavljajo najnižjo spodnjo levo točko in najvišjo zgornjo desno točko objekta. V primeru točke seveda uporabimo koordinate točke same. S temi koordinatami lahko nato ustrezno premaknemo trenutni pogled nad risbo v programu BricsCAD. Spodnja leva in zgornja desna točka nam omogočata, da okoli celotnega objekta postavimo navidezni pravokotnik, ki predstavlja trenutni pogled nad risbo. V logiki svoje aplikacije z uporabo teh dveh točk naprej izračunamo centralno točko navideznega pravokotnika in izračunamo še širino ter višino tega pravokotnika. Preostane nam samo še klic funkcije "acedSetCurrentView", ki nam omogoča spreminjanje trenutnega pogleda nad risbo in je del BricsCAD-ovega SDK-ja. Kot parameter ji posredujemo svoj izračunani navidezni pravokotnik in pogled nad risbo se nam spremeni v skladu s posredovanim parametrom.

Naslednja slika (Slika 5.6) prikazuje celotno podobo končnega rezultata delovanja naše aplikacije v programu BricsCAD. Uporabnik je v tem primeru izbral in uvozil datoteko "`LJ_Center_Bezigrad`". Naša aplikacija je na podlagi imena datoteke ustvarila in dodala sloj v samo risbo ter v uporabniški vmesnik dodala zavihek z istim imenom. Nato je iz izbrane datoteke pridobila podatke o objektih. Geometrijske podatke, katerih tip je v tem primeru poligon, je dodala na sloj in hkrati tudi v samo risbo. Istočasno je aplikacija še uredila in pripravila tabelo za prikaz atributnih podatkov ter jo dodala v naš uporabniški vmesnik. Uporabnik je nato v risbi izbral določene objekte in naša aplikacija je izpostavila atributne podatke izbranih objektov.




Slika 5.6: Prikaz delovanja celotnega sistema.

### 5.2.3 Izdelava namestitve

Aplikacijo smo želeli zaključiti in združiti v neko celoto, saj je bila razvita z namenom, da bi jo lahko projektanti uporabljali v svojem delovnem procesu. Pri predstavitvi aplikacije uporabniku je zelo pomemben že prvi vtis. Sistem, kjer mora projektant naprej aplikacijo razširiti iz stisnjene arhivske datoteke in nato poganjati iz neke mape ne daje učinkovite podobe. S tem namenom smo dodali še klasičen sistem za namestitev in odstranitev aplikacije ter zadostili zahtevi za preprosto izvedbo le-tega. Za vpeljavo tega sistema smo uporabili sistem NSIS, odprtokodni skriptni sistem za ustvarjanje namestitvenih programov v operacijskem sistemu Windows.

Instalacijski skripti smo naprej dodali vtičnik MUI, tj. vtičnik za moderni grafični uporabniški vmesnik, ki sistemu za namestitev doda standardni čarovnik za namestitev. Poleg tega se je sistemu dodal še dodatni vtičnik, ki skrbi za beleženje datotek, ki se prenesejo med namestitvijo. Njegov namen pride do izraza pri odstranitvi aplikacije, kjer zelo poenostavi delo. Med izvajanjem namestitve poskrbimo za posodobitev registra s podatki o aplikaciji, ter na namizje in orodno vrstico dodamo ustrezne bližnjice za zagon aplikacije in odstranitev aplikacije.

```
395
396 Section "SHPReader install"
397   SectionIn 1 2 3 RO
398   SetShellVarContext all
399
400   ${If} ${RunningX64}
401     SetRegView 64 ;set the registry view to x64
402   ${Else}
403     SetRegView 32 ;set the registry view to x86
404   ${EndIf}
405
406   WriteRegStr "${PRODUCT_UNINST_ROOT_KEY}" "${PRODUCT_UNINST_KEY}" "DisplayName" "${PRODUCT_NAME}"
407   WriteRegStr "${PRODUCT_UNINST_ROOT_KEY}" "${PRODUCT_UNINST_KEY}" "ProductLocation" "$INSTDIR"
408   WriteRegStr "${PRODUCT_UNINST_ROOT_KEY}" "${PRODUCT_UNINST_KEY}" "UninstallString" "$\"$INSTDIR\uninstall.exe$\""
409   WriteRegStr "${PRODUCT_UNINST_ROOT_KEY}" "${PRODUCT_UNINST_KEY}" "Publisher" "${PRODUCT_PUBLISHER}"
410   WriteRegStr "${PRODUCT_UNINST_ROOT_KEY}" "${PRODUCT_UNINST_KEY}" "DisplayVersion" "${VERSION}"
```

Slika 5.7: Izsek instalacijske skripte, ki zapiše osnovne podatke aplikacije v register.

Pred samo namestitvijo aplikacije s skripto v registru računalnika preverimo obstoj starejše verzije naše aplikacije. ČE skripta zazna, da uporabnik že ima predhodno nameščeno aplikacijo mu samodejno ponudi opcijo

izbiro odstranitve trenutne verzije aplikacije. S tem smo poenostavili težavo nadgradnje aplikacije tako, da je potrebno pred uporabo nove verzije najprej v celoti odstraniti staro verzijo. Naša aplikacija je v končnem pomenu vtičnik za program BricsCAD in če ta program ni prisoten na uporabnikovem računalniku, tudi naša aplikacija ne bo delovala. S tem namenom v skripti pred samo namestitvijo še preverimo prisotnost programa BricsCAD in prekinemo namestitev v primeru, da ta program ni prisoten.

```
247  ClearErrors
248
249    ReadRegStr $0 HKCU "SOFTWARE\Bricsys\BricsCAD" "CURVER"
250    IfErrors 0 +4
251      MessageBox MB_OK $ErrorBox_1
252      Abort
253      Return
254
255    ReadRegStr $1 HKCU "SOFTWARE\Bricsys\BricsCAD\$0" "CURVER"
256    IfErrors 0 +4
257      MessageBox MB_OK $ErrorBox_2
258      Abort
259      Return
260
261    SetRegView 32 ;set the registry view to x86
262    ReadRegStr $BricsCAD_LOCATION HKLM "SOFTWARE\Bricsys\BricsCAD\$0\$1" "InstallDir"
263    IfErrors 0 +4
264      MessageBox MB_OK $ErrorBox_3
265      Abort
266      Return
```

Slika 5.8: Izsek instalacijske skripte, ki preveri prisotnost programa BricsCAD.

# Poglavje 6

# Sklepne ugotovitve

Namen diplomskega dela je bil razvoj aplikacije, ki bo premostila vrzel v povezanosti med cenovno ugodnejšimi alternativnimi programi CAD in podatki iz sistema GIS. Ta cilj smo uspešno dosegli, kajti aplikacija, ki je rezultat tega diplomskega dela, naredi velik korak v smer takšne povezave. Projektantom oziroma uporabnikom omogoča dodajanje objektov iz datoteke shp v svoje risbe in pregled atributnih podatkov teh objektov. S tem jim omogočimo boljši vpogled v trenutno stanje, kadar načrtujejo gradnjo novih objektov (cest, hiš, kanalizacij ...) v obstoječem okolju ali kadar načrtujejo spremembo že obstoječih objektov.

Aplikacija v trenutni različici uporabnikom ponuja konkretno podlago za delo na dejanskem projektu, čeprav ne moremo spregledati določene nadgradnje, ki bi v veliki meri povečala vrednost aplikacije:

- Prva nadgradnja, ki bi v največji meri povečala vrednost aplikacije, je možnost dodajanja, urejanja in brisanja geometrije kot tudi atributnih podatkov objektov v datotekah shp. Ta nadgradnja bi skupaj z že dosedanjimi funkcionalnostmi uporabnikom omogočala vse prave funkcije sistema GIS.

- Naslednji sklop nadgradenj bi uporabnikom dodal možnost filtriranja objektov in njihovih atributnih podatkov. Naprej bi uporabniku omogočili izbor objektov, ki jih želi uvoziti v risbo. To je koristno

predvsem v primeru, kadar uporabnik že predhodno ve, katere objekte potrebuje. Dodatna nadgradnja bi uporabniku tukaj dodala možnost filtriranja. To bi pomenilo prikazovanje/skrivanje objektov, potem ko so že dodani v risbo glede na določeno vrednost atributnih podatkov.

- V svoji aplikaciji smo se omejili na branje datotek shp. Vendar podatki GIS niso zapisani samo v tej obliki. Aplikacijo bi bilo smiselno nadgraditi tudi z možnostjo branja drugih vrst zapisa teh podatkov, od datotek gml, xml, sdf do neposredne povezave na podatkovne baze, kot so SQL ali Oracle Spatial. Vključitev takšne nadgradnje bi poleg možnosti urejanja podatkov predstavljalo izredno dodatno vrednost in uporabnost naše aplikacije. Vendar bi vpeljava te nadgradnje zahtevala veliko časa za vsako vrsto zapisa podatkov.

- Pozabiti pa ne smemo na manjše izboljšave, ki bi na primer omogočale branje celotne mape in ne vsake datoteke posebej kot trenutno deluje aplikacija, itd.

Omeniti moramo še dejstvo, da smo pri razvoju aplikacije uporabili dva različna programska jezika. Programski jezik C++ je skrbel za logiko same aplikacije, medtem ko je programski jezik JavaScript (v povezavi s tehnologijami HTML in CSS) poskrbel za grafični vmesnik aplikacije. Kot smo že predhodno omenili, smo želeli preveriti, na kakšen način lahko ta dva različna principa programiranja sodelujeta. Vseeno menimo, da bi bilo za kakovosten razvoj resnejše aplikacije bolj smiselno, če se osredotočimo samo na en programski jezik, ki je v primeru naše aplikacije programski jezik C++. S tem bi sicer izgubili prednosti drugega programskega jezika, toda iz razvoja in delovanja aplikacije bi odstranili slabosti in omejitve drugega jezika. S to odločitvijo tudi zmanjšamo zahtevnost vzdrževanja aplikacije.

Informativno smo preverili, ali na tržišču že obstajajo podobne aplikacije kot je naš SHPReader. Ugotovili smo, da so za program BricsCAD dejansko na voljo že nekatera orodja, ki ponujajo podobne funkcionalnosti. Vendarle so ta orodja ali zelo primitivna, saj ponujajo golo branje in prikazovanje

vektorskega zapisa oziroma geometrije objektov, izpustijo pa atributne podatke, ali pa so vpeta v večje programe, kar pod vprašaj postavlja njihovo cenovno dostopnost. Menimo, da bi lahko SHPReader večje število projektantov s pridom izkoristilo v procesu svojega dela že v obstoječi obliki, brez prej omenjenih možnosti za nadgradnje.

# Literatura


[1] BricsCAD. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/BricsCAD. [Dostopano oktober 2015].

[2] Computer-aided design. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Computer-aided_design. [Dostopano oktober 2015].

[3] CSS. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Cascading_Style_Sheets. [Dostopano oktober 2015].

[4] D. Flanagan, *JavaScript: The Definitive Guide, Sixth Edition*. O'Reilly Media, Beijing [etc.]: O'Reilly, 2011, str. 1–3

[5] FDO Developer's Guide -> FDO Concepts -> Data Concepts. [Online]. Dosegljivo: http://www.osgeo.org/files/fdo/docs/FDG_FDODevGuide/files/WS7106c181349dd8d0913fe2105df83c358-7ffe.htm. [Dostopano oktober 2015].

[6] FDO Developer's Guide -> OSGeo FDO Provider for SHP -> What Is FDO Provider for SHP?. [Online]. Dosegljivo: http://www.osgeo.org/files/fdo/docs/FDG_FDODevGuide/files/WS7106c181349dd8d01ef8cf3105dfb865a4-8000.htm. [Dostopano oktober 2015].

[7] Geographic information system. [Online]. Dosegljivo: http://wiki.gis.com/wiki/index.php/GIS. [Dostopano oktober 2015].

[8] GIS (geographic information system). [Online]. Dosegljivo: http://education.nationalgeographic.com/education/encyclopedia/geographic-information-system-gis/. [Dostopano oktober 2015].

[9] HTML. [Online]. Dosegljivo: http://en.wikipedia.org/wiki/HTML. [Dostopano oktober 2015].

[10] JQuery. [Online]. Dosegljivo: http://jquery.com. [Dostopano oktober 2015].

[11] JQuery UI. [Online]. Dosegljivo: http://jqueryui.com. [Dostopano oktober 2015].

[12] Mercurial. [Online]. Dosegljivo: https://www.mercurial-scm.org/about. [Dostopano oktober 2015].

[13] NSIS - zgodovina. [Online]. Dosegljivo: http://en.wikipedia.org/wiki/Nullsoft_Scriptable_Install_System. [Dostopano oktober 2015].

[14] NSIS. [Online]. Dosegljivo: http://nsis.sourceforge.net. [Dostopano oktober 2015].

[15] NWS Birmingham Geographic Information Systems Data. [Online]. Dosegljivo: http://www.srh.noaa.gov/bmx/?n=gis. [Dostopano oktober 2015].

[16] OSGeo FDO. [Online]. Dosegljivo: http://fdo.osgeo.org/overview.html. [Dostopano oktober 2015].

[17] Qt (software). [Online]. Dosegljivo: http://en.wikipedia.org/wiki/Qt_(software)#Qt_4. [Dostopano oktober 2015].

[18] Qt/MFC Migration Framework. [Online]. Dosegljivo: http://docs.huihoo.com/qt/solutions/4/qtwinmigrate/index.html/. [Dostopano oktober 2015].

[19] Shapefiles -> About Shapefiles. [Online]. Dosegljivo: http://wiki.openstreetmap.org/wiki/Shapefiles. [Dostopano oktober 2015].

[20] TortoiseHg. [Online]. Dosegljivo: http://tortoisehg.bitbucket.org/about.html. [Dostopano oktober 2015].

[21] Visual Studio. [Online]. Dosegljivo: http://msdn.microsoft.com/en-us/library/fx6bk1f4(v=vs.100).aspx. [Dostopano oktober 2015].

[22] Visual C++. [Online]. Dosegljivo: http://msdn.microsoft.com/en-us/library/60k1461a(v=vs.100).aspx. [Dostopano oktober 2015].